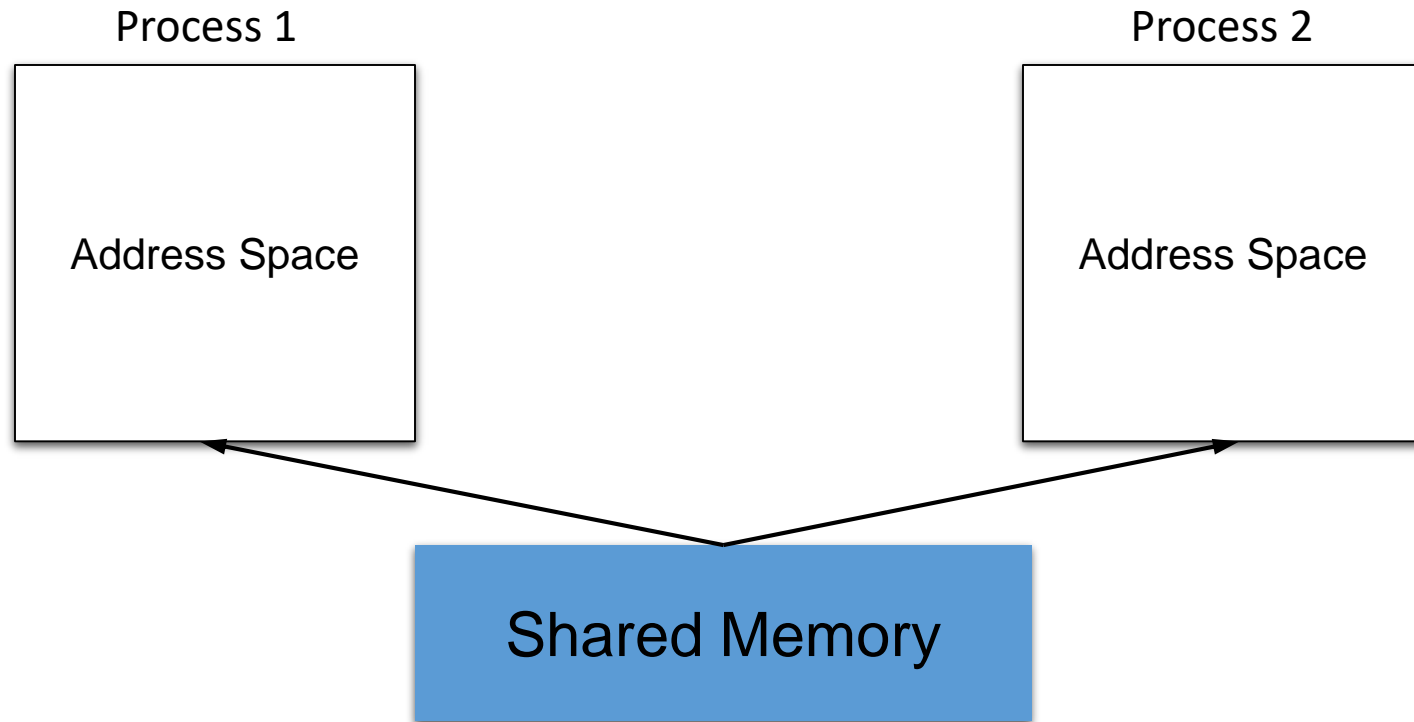


# Shared Memory

- Mecanismo altamente eficiente de comunicación entre procesos.



# Problema de acceso a recursos compartidos

Proceso 1	Proceso 2
<code>a = (*pdata) ;</code>	<code>b = (*pdata) ;</code>
<code>a++;</code>	<code>b--;</code>
<code>*pdata = a;</code>	<code>*pdata = b;</code>

- Necesidad de utilizar algún mecanismo de sincronización de acceso



# POSIX shared Memory

```
int shm_open(const char *name,  
             int oflag,  
             mode_t mode);
```



```
int shm_unlink(const char *name);
```



```
void *mmap(void *addr, size_t length,  
           int prot, int flags,  
           int fd, off_t offset);  
int munmap(void *addr, size_t length);
```

# POSIX semaphores

```
#include <semaphore.h>
```



```
int sem_init(sem_t *sem, int pshared,  
             unsigned int value);
```

```
int sem_wait(sem_t *sem);
```



```
int sem_trywait(sem_t *sem);
```



```
int sem_timedwait(sem_t *sem,  
                  const struct timespec *absTime);
```



```
int sem_post(sem_t *sem);
```

```
int sem_destroy(sem_t *sem);
```

# POSIX messages queue

- Mecanismo eficiente de IPC.
- Se crean y abren con **mq\_open**. Esta función retorna un message queue descriptor (**mqd\_t**).
- Cada cola es identificada por un nombre ("**msq**")
- Los mensajes son transferidos utilizando **mq\_send** y **mq\_receive**.
- Se cierran con **mq\_close**.
- Se destruyen con **mq\_unlink**.
- **mq\_getattr** y **mq\_setattr** para acceso a atributos.
- Cada mensaje tiene una prioridad asociada.

# POSIX messages queue

```
#include <mqueue.h>
```

```
mqd_t mq_open(const char *name, int oflags);  
mqd_t mq_open(const char *name, int oflags,  
               mode_t mode, struct mq_attr *attr);
```

O\_RDONLY

O\_WRONLY

O\_RDWR

O\_NONBLOCK -> operaciones bloqueantes salen con EAGAIN

O\_CREAT -> se deben aplicar los otros 2 parámetros

O\_EXCL

# POSIX messages queue

```
#include <mqueue.h>
```

```
int mq_send(mqd_t q, const char *ptr,  
            size_t len, unsigned msg_prio);
```

```
int mq_timedsend(mqd_t q, const char *ptr,  
                 size_t len, unsigned msg_prio,  
                 const struct timespec *absT);
```

```
ssize_t mq_receive(mqd_t q, const char *ptr,  
                  size_t len, unsigned *pMsg_prio);
```

```
ssize_t mq_timedreceive(mqd_t q, const char *ptr,  
                        size_t len, unsigned *pMsg_prio,  
                        const struct timespec *absT);
```

# POSIX messages queue

```
#include <mqueue.h>
```

```
int mq_getattr(mqd_t mqdes, struct mq_attr *attr);
```

```
int mq_setattr(mqd_t mqdes, struct mq_attr *newattr,  
               struct mq_attr *oldattr);
```

```
struct mq_attr {  
    long mq_flags;    /* Flags: 0 or O_NONBLOCK */  
    long mq_maxmsg;   /* Max. # of messages on queue */  
    long mq_msgsize;  /* Max. message size (bytes) */  
    long mq_curmsgs;  /* #of messages currently in q */  
};
```



# POSIX messages queue

```
#include <mqueue.h>
```

```
int mq_notify(mqd_t mqdes,  
              const struct sigevent *sevp);
```

Se registra o deregistra para la recepción asincrónica de notificaciones cuando un mensaje llega a una cola vacía.